# Efficient 3D Binary Image Skeletonization

Son Tran and Liwen Shih
*Computer Engineering, University of Houston - Clear Lake (UHCL)*
*shih@uhcl.edu*

## Abstract

*Image Skeletonization promises to be a powerful complexity-cutting tool for compact shape description, pattern recognition, robot vision, animation, petrography pore space fluid flow analysis, model/analysis of bone/lung/circulation, and image compression for telemedicine. The existing image thinning/skeletonization techniques using boundary erosion, distance coding, and Voronoi diagram are first overviewed to assess/compare their feasibility of extending from 2D to 3D. An efficient distance-based procedure to generate the skeleton of large, complex 3D images such as CT, MRI data of human organ is then described. The proposed 3D Voxel Coding (3DVC) algorithm, is based on Discrete Euclidean Distance Transform. Instead of actual distance, each interior voxel (3D pixel) in the 3D image object is labeled with an integer code according to its relative distance from the object border for computation efficiency. All center voxels, which are the furthest away from the object border, are then collected and thinned to form clusters. To preserve the topology of the 3D image object, a cluster-labeling heuristic is then applied to order the clusters, and to recursively connect the next nearest clusters, gradually reducing the total number of disjoint clusters, to generate one final connected skeleton for each 3D object. The algorithm provides a straightforward computation which is robust and not sensitive to noise or object boundary complexity. Because 3D skeleton may not be unique, several application-dependent skeletonization options will be explored for meeting specific quality/speed requirements, and perhaps to incorporate automatic machine intelligence decisions. Parallel version of 3DVC is also introduced to further enhance skeletonization speed.*

**Keyword**: *3D Image Skeleton, Euclidean Distance Transform, Parallel Algorithm, Heuristics*

Acknowledgement: Partially sponsored by TX HECB ARP/ATP grants, the project started when working to cut complexity in University of Houston professor Kishore Mohanty's pioneer work on oil/water/gas flow analysis through petrography pore spaces.

## 1. 2D and 3D Image Skeletonization

The generation of a digital image skeleton is often one of the first processing steps taken by a computer vision system when attempting to extract features from an object in an image. Due to its compact shape representations, image skeletonization has been studied for a long time in computer vision, pattern recognition and Optical Character Recognition. It is a powerful tool for intermediate representation for a number of geometric operations on solid models. An image skeleton is presumed to represent the shape of the object in a relatively small number of pixels/voxels, all of which are, in some sense, structural and therefore necessary. The *skeleton of an object* is, conceptually, defined as the locus of center voxels in the object. Unfortunately, no generally agreed upon definition of a digital image skeleton exists. Of the literally hundreds of papers on the subject of thinning which are in print, the vast majority are concerned with the implementation of a variation on an existing thinning method, where the novel aspects are related to the performance of the algorithm [11][14][17]. But for all definitions, at least 4 requirements below must be satisfy for skeleton objects:

- Centeredness satisfaction: Skeleton is geometrically centered within the object boundary or as close as possible.
- Connectivity preservation: The output skeleton should have the same connectivity as the original object and should not contain any background element.

- Topology must remain constant
- As thin as possible: 1-voxel thin is the requirement for a 2D skeleton, and in 3D, as thin as possible.

## 2. 3D Image Skeletonization Challenges

There are a variety of methods proposed for image skeleton extraction in the literature. In general, they can be classified in 3 categories:

### 2.1. Boundary peeling/erosion

This method iteratively peels off the boundary voxel-by-voxel, layer-by-layer from outside to inside of the object where removal does not affect the topology of the object. This is a repetitive, time-intensive process of testing and deletion each voxel. The difficulty of this method is that the set of rules defined for removing voxel is dependent highly on the type of image and that different set of rules will be applied for different type of images. However, this method is good for connectivity preservation. [1][2][3]

### 2.2. Euclidean Distance coding/transform

This method is based on the distance of each voxel to the boundary, and tries to directly extract skeleton voxels by finding the local maximum voxels (voxels in the center of object). The distance coding is based on Euclidean distance or the *approximation* to Euclidean distance. This method is faster and can be done in *high degree of parallelism*, however the output is not guaranteed to preserve connectivity [5][6][10][12].

The main problem of this method is connectivity. Niblack et al. [13] provided a solution in 2D. In order to connect local maxima using uphill climbing rules, they added saddle points, which are local minima along a skeleton, to the skeletal point set. However, direct extension of this algorithm to 3D is difficult because there are not necessarily unique sequences of voxels around a given voxel. Helman and Hesselink [15] detected saddle points in a 2D vector field by computing the eigenvalues of the Jacobian matrix of the field. The accuracy of saddle points depends on the accuracy of the vector field generated from a distance transform. In 3D, noisy, complex data would have too many voxels been taken as saddle points, resulting in either erroneous skeleton parts or very thick skeletons. Y. Zhou [5] recently proposed a fast, template-based skeleton and centerline extraction method, which is based on 2 coding operations: boundary-seeded (BS) voxel coding and single point seeded (SS) voxel coding. From BS, all local maxima will be detected to generate center clusters, from SS, all center clusters will be connected to generate skeleton using Local Maximum Path concept. His algorithm works well on MRI and CT images, which have pipe or torus shape. For other shapes such as cube, polygon block, the result is a *plane* instead of a single line.

This method takes 3 steps:

- Generate the minimum distance field of the object.
- Detect all local maxima in term of distance field to generate center clusters. From now on, the object is considered as a set of clusters instead of a set of voxels.
- Reconnect all clusters to generate the skeleton.

### 2.3 Distance transform - Voronoi Diagram

This method [7][8][9], theoretically, guarantees connected skeletons, is best suited for polygonally defined objects. However, real image data such as CT, MRI with large amount of voxels and noise, cause Voronoi diagram to be very dense and the computation time to be expensive because they generate too many nodes and lines. Computing by this method is also impossible for small computer with small memory since the whole image data (with 256 x 256 x 128) requires at least 8M of memory to load the image.

So far, no algorithm works well with a general image, nor satisfies the accuracy and time performance requirements. Voronoi method produces good result, but the computing time is large, Border erosion method is faster than Voronoi, but is also less accurate. Distance Transform method gives the best performance, but also is difficult in implementation and programming to preserve connectivity. Connectivity step (or how to connect all clusters) is also the most difficult step that many authors have proposed algorithms to solve but ended up either too complicated or inefficient. We are proposing a simple and efficient algorithm to solve this problem as discussed in later sections. Most notations and terms used in this project are kept the same as used by authors in [1][5][16].

## 3. Center Clusters and Connectivity

### 3.1. Binary Image

The term binary image refers to the image with all voxels carrying the value of 0 or 1. A 3D binary image with size $M$ x $N$ x $P$ can be represented as a 3D array, IMG $\equiv \{v(i, j, k) \mid 1 \le i \le M, 1 \le j \le N, 1 \le k \le P\}$ with $M, N, P$ being positive integers and $v(i, j, k) \in \{0, 1\}$.

In image visualization, 0's voxel will be displayed as black and 1's voxel will be white.

### 3.2. Euclidean Distance

Euclidean Distance refers to the actual distance between voxels in the image:

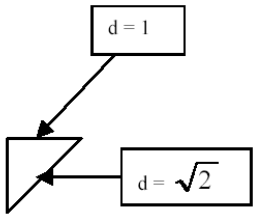



**Figure 1: Euclidean distance**

Figure 2 shows an example of the distance from every voxels to a given voxel (shaded):

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 1 | $\sqrt{2}$ | $\sqrt{5}$ | $\sqrt{10}$ |
| 2 | $\sqrt{5}$ | $\sqrt{8}$ | $\sqrt{13}$ |
| 3 | $\sqrt{10}$ | $\sqrt{13}$ | $\sqrt{18}$ |
| 4 | $\sqrt{17}$ | $\sqrt{20}$ | 5 |

**Figure 2: Example of Euclidean distances**

### 3.3. Geometrical Connectivity

Euler's formula provided a means for checking the connectivity for 3D objects. For each single closed netted surface, if $n$ denotes the number of nodal points in the net, $f$ denotes the number of faces, and $e$ denotes the number of edges, we have:

$$n - e + f = 2$$

More generally, when handles or cavities are present in object, the formula becomes:

$$n - e + f = 2 - 2 * h,$$

where $h$ is the number of handles.

For the entire 3D image, one may define the connectivity number $N$ as

$$N = \sum(2 - 2h_i)$$

In order to preserve the connectivity, the value $N$ should not be changed during skeletonization as in [3].

### 3.4. Notations used in this project

- *Background voxel*: A voxel with value = 0 black.
- *Object voxel*: Voxel with value = 1 white.
- *Boundary voxel*: Voxel with at least one of its neighbor being a background voxel.
- *Outside Voxel*: Voxel with all of its neighbors = 0.
- *Inside Voxel*: Voxel with all of its neighbors = 1.
- *F-connected*: Two voxels $P_i$ and $P_j$ are called *F*-connected if they shared a *face*. And $P_j$ is an ***F-neighbor*** of $P_i$. There are 6 *F*-connected voxels of a given voxel

**Figure 3: *F*-connected voxels**

- *E-connected*: Two voxels $P_i$ and $P_j$ are called *E*-connected if they share an *edge*. And $P_j$ is an ***E-neighbor*** of $P_i$. There are 12 *E*-connected voxels of a given voxel

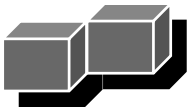


**Figure 4: *E*-connected voxels**

- *V-connected*: Two voxels $P_i$ and $P_j$ are called *V*-connected if they share a *vertice*. And $P_j$ is a ***V-neighbor*** of $P_i$. There are 8 *V*-connected voxels of a given voxel.

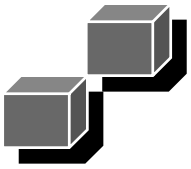


**Figure 5: V-connected voxels**

- *Definitions:*

  *Distance Value* - Is the value assigned to a voxel based on the approximate Euclidean distance of that voxel from the border. A voxel is call *n*-voxel if it's distance value = *n*.

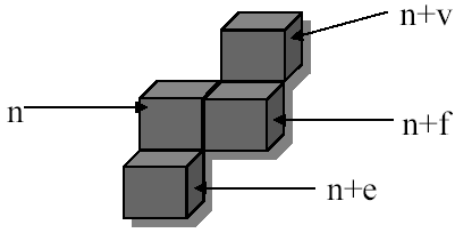



**Figure 6: Distance code progression**

*Coding Vector or Matrix* – [f,e,v], eg., [1,2,3].

*Local Maximum Voxel* - Is the voxel that has the distance value not less than those of all its *E/F*-neighbors [4].

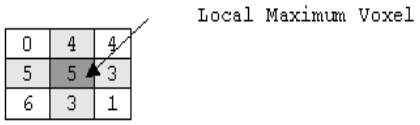



**Figure 7. Example of Local Maximum Voxel**

*Cluster* - Is defined as a set of *F/E/V*-connected local maximum voxels that have the same distance value.

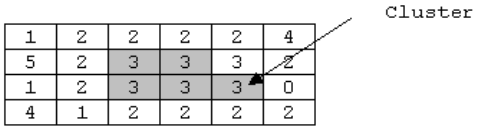



**Figure 8. Example of cluster**

*The Shortest Path* $P \equiv \{p_i \mid i = 1 \rightarrow n\}$ is a sequence of voxel $p_i$ such that:

i. Each $p_i$ has at least one *F*-Neighbor with a distance greater than that of $p_i$

ii. Voxel $p_1$ and $p_n$ have only 1 neighbor – they are 2 ends of the path, voxel $p_i$ $(1 < i < n)$ has exactly 2 neighbors.

iii. $p_i$ and $p_{i+1}$ are connected, $p_i$ and $p_{i+2}$ are disconnected.

In this paper, the shortest path is used to connect 2 separate clusters and will be discussed in more detail later.

## 4. 3D Voxel Coding and Cluster Labeling

Our algorithm proposed here works well in 2D and is also extendable to 3D. The method belongs to the second category, which uses a discrete coding method of Euclidean transform. The reason to use this method is because when we process large image with millions of voxels, the accuracy of a few voxels does not affect the whole result, but the reduction in process time is important. Beside the four main requirements mentioned above for skeletons, some more characteristics are also considered in the project:

- Straightforward computation: This requirement is to speed up the computation time.
- Work well with many kinds of images.
- Efficient object hole detection: Some application requires designated hole detection.
- No sensitivity to object boundary complexity, where Noise removal / Image enhancement is required.
- Visualization.

The proposed 3D Voxel Coding and Cluster Labeling skeletonization method includes 8 steps:

1. Preprocess image data and remove noise if necessary.
2. Detect the boundary.

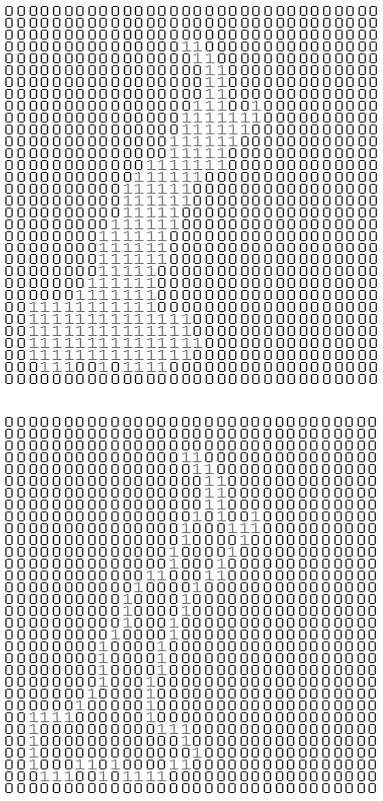


**Figure 9: Border detect**

3. Determine minimum distance field for the object by using voxel-coding method (Figure 10).

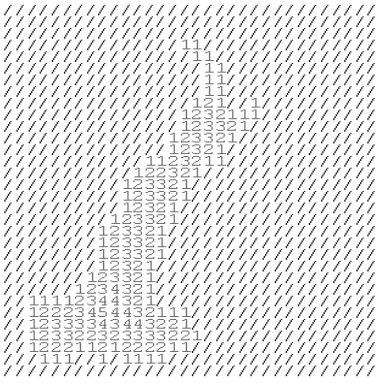


**Figure 10: Voxel-coding**

4. Generate clusters based on result from step 3.

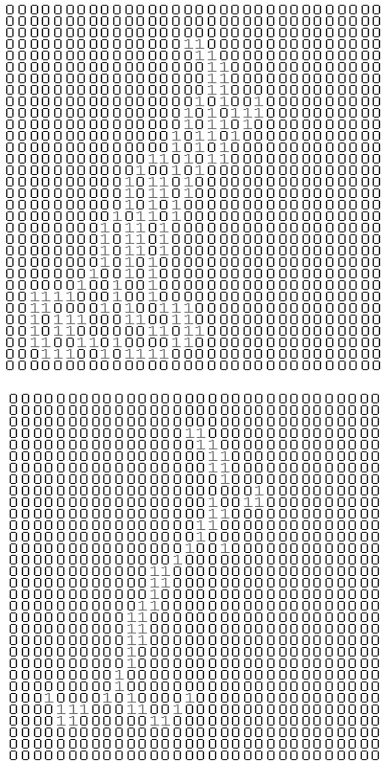
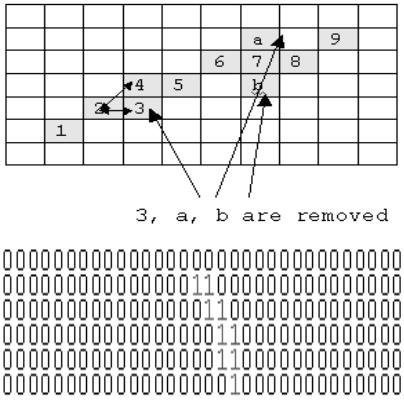


**Figure 11: Cluster generation**

5. Refine the clusters to be as thin as possible: Each voxel should have only 2 neighbors except end voxels. No voxels share same neighbors except branch voxel. Keep only voxels on the shortest path.



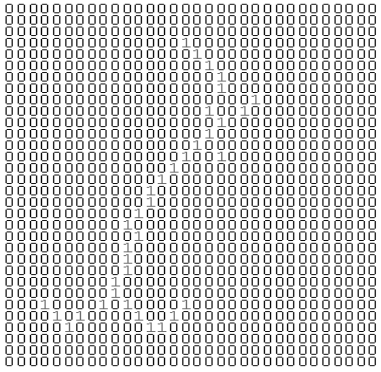



**Figure 12: Cluster thinning**

6. Detect for holes in the object.
7. Connect all clusters to generate the skeleton. Clusters are randomly ordered and labeled with sequential numbers from #2, #3, and so on (0 and 1 are reserved for the background and the thinned clusters) as the example in Figure 13. Starting at a seed cluster, say cluster #2 (or multiple seed clusters in parallel), shell (expand layer by layer) cluster #2 until touching the next nearing cluster, say #3. Connect both touching clusters #2 and #3 using shortest path, and then name the newly connected cluster with the larger cluster number as in Figure 14. Snowballing the clusters connection recursively until all clusters are connected to one final cluster #5. Note that the shelling layer voxels are temporary labeled with numbers larger than the largest cluster label number (#6,7,8,9 etc. > #5) to avoid confusion between cluster and non-cluster voxels.

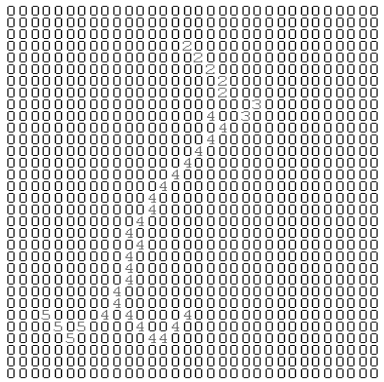


**Figure 13: Cluster labeling #2,3,4,5**

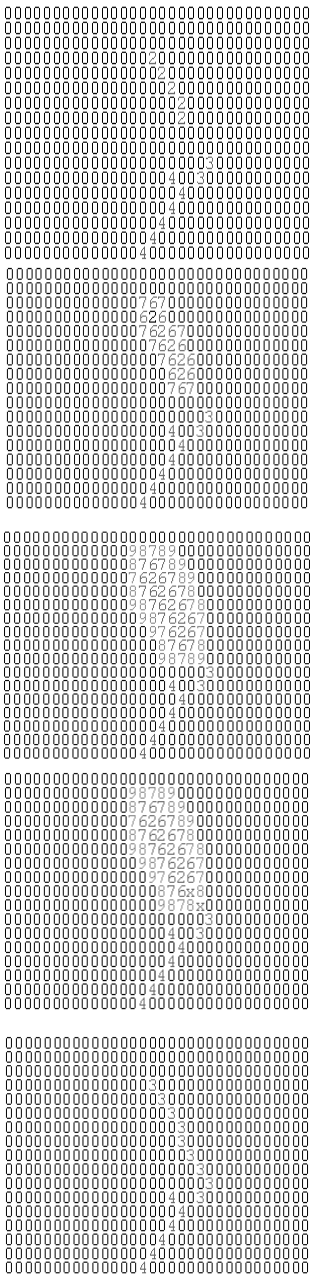


**Figure 14: Extend cluster #2 by shelling until touching to join the nearest cluster #3**

8. Output skeleton to a file and display the result.

The steps above can be done either in sequential or parallel. In parallel, the first three functions can be split out to many processes because each process can work independently. The fourth function can be divided to the maximum of processes equal to the number of clusters found. The fifth function can be divided to (the number of clusters / 2) processes. The level of parallelism of this method is also higher than previous methods as discussed above.

## 5. Efficient 2D and 3D Skeletonization

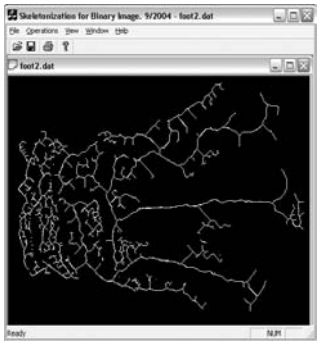
We have tested many data sets with this algorithm, from a simple 2D image to complex, large MRI image. The result is good in both 2D and 3D, see Figure 15. The 2D display is integrated in the windows environment tool developed in C++ and the 3D images are displayed using Matlab version 5.3 or later.

The execution speed in sequential version also surpasses all current algorithms as shown in Table 1:

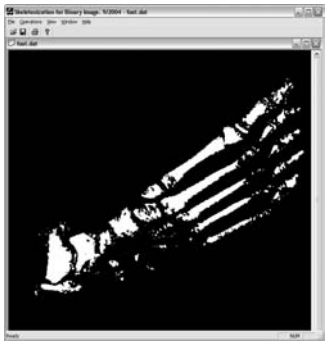
**Table 1. Skeletonization speed increase**

| Image Size | Boundary Peeling time | 3DVC (second) |
|---|---|---|
| 64 x 64 | 0.03 | 0.02 |
| 256 x 256 | 0.47 | 0.19 |
| 512 x 512 | 2.98 | 0.61 |
| 1024 x 1024 | 18.98 | 2.64 |
| 64 x 64 x 64 | | 1.59 |
| 128 x 128 x 27 | | 4.52 |
| Test on Pentium 500 MHz, 256 M RAM | | |

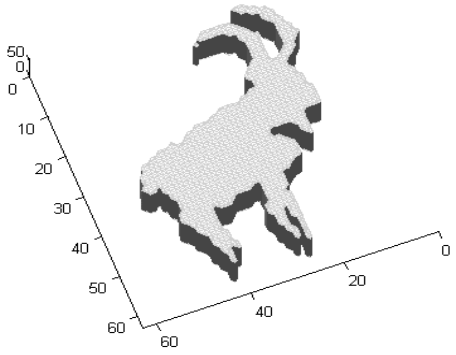
**a. 2D skeleton example (256 x 256)**

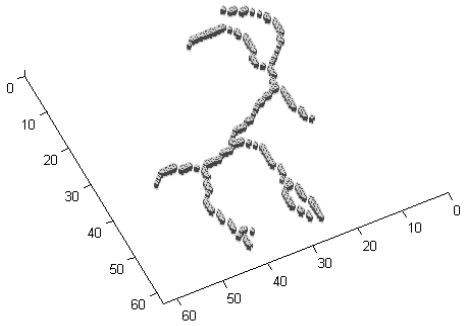


**b. 2D skeleton example**

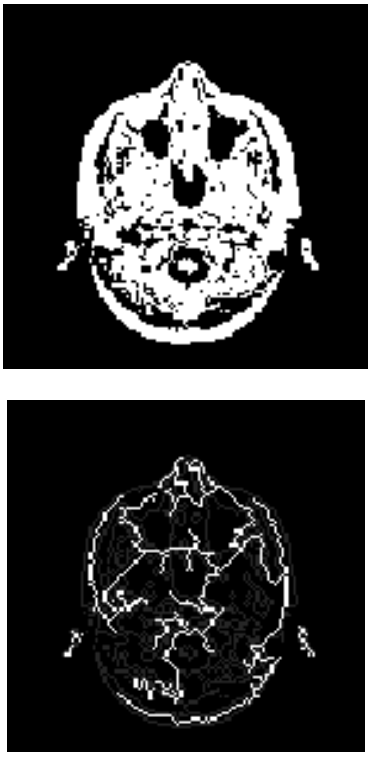


**c. 2D skeleton example**

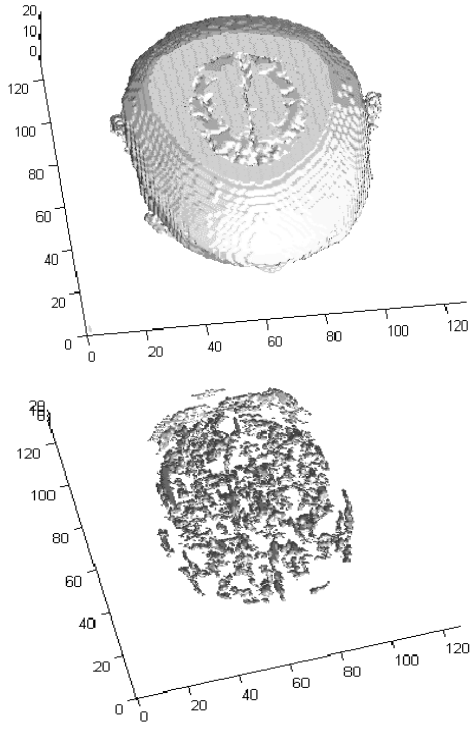



**d. 3D skeleton example (64 x 64 x 64)**



**e. 1 slice of MRI image (128 x 128 x 27)**



**f. The whole MRI image**

**Figure 15: Results of 2D/3D skeleton**

## 6. Intelligent Image Skeleton extractor

The Algorithm developed above works well with many kinds of image data set including MRI, CT and Polygon Object. But the *Cluster Refinement* routine can be modified depending on the shape of the objects to satisfy particular requirement. Certain images, with the pipe/round shape like cylinder, torus, CT, MRI, generate a smooth skeleton without refinement step. Others need to be smoothed and refined many times. It is difficult to find a general rule that satisfies all kinds of image. A promising solution would be to incorporate automatic application-dependent image thinning decisions based on machine intelligence.

## 7. Improved Skeletonization Performance

We have presented the Euclidean Distance Transform Based skeleton extraction for skeletons consisting of clusters and paths. Current Skeletonization methods are overviewed and compared. The advantage of this 3D Voxel Coding method is the high performance compared to other methods. The disadvantage is the lost of connectivity of the skeleton. The efficient cluster labeling heuristic algorithm proposed overcomes that disadvantage; generates a smooth, thin, connected skeleton that satisfied most of 3D data sets. Some limitations of this project are also discussed for further investigation and improvement.

## 8. Reference


[1] Y.F. Tsao and K.S. Fu, "A Parallel Thinning Algorithm for 3-D Pictures", *CGIP* no. 17, 1981, pp. 315-331.

[2] C.M. Mao and M. Sonka, "A Fully Parallel 3D Thinning Algorithm and Its Applications", *Computer Vision and Image Understanding*, vol. 64, no. 3, 1996, pp. 420-433.

[3] S.Lobregt, P.W. Verbeek, and F.C.A. Groen, "Three-Dimensional Skeletonization: Principle and Algorithm", *IEEE Transaction on PAMI,* vol. 2, 1980, pp.75-77.

[4] C. Arcelli and G. Saniti di Baja, "Finding Local Maxima in a Pseudo-Euclidean Distance Transform", *Computer Vision, Graphics and Image Processing*, vol. 43, 1988, pp. 361-367.

[5] Y. Zhou, A. Kaufman, and A.W. Toga, "3D skeleton and Centerline Generation Based on an Approximate Minimum Distance Field", *The Visual Computer*, vol.14, no. 7, 1998, pp 303-314.

[6] L. Dorst, "Pseudo-Euclidean Skeletons", *Proc. Eighth Int'l Conf. Pattern Recognition,* 1986, pp. 286-289.

[7] R.L. Ogniewicz and O. Kubler, "Hierarchic Voronoi Skeletons", *Pattern Recognition*, vol. 28, no. 3, 1995, pp. 343-359.

[8] R.L. Ogniewicz and M. Ilg, "Voronoi skeletons: Theory and applications", *Proc. Conf. On CVPR*, 1992, pp. 63-69.

[9] R.L. Ogniewicz, "Skeleton-Space: a Multiscale Shape Description Combining Region and Boudary Information", *Proc. CVPR*, 1994, pp. 746-751.

[10] G. Borgefors, "Distance Transformation on Digital Images", *Computer Vision Graphics Image Processing*, vol 34, 1986, pp. 344-371.

[11] Itoh T, Yamaguchi Y, Koyamada K, "Volume thinning for automatic isosurface propagation. *IEEE Proceeding of Visualization'96*, San Francisco, CA, Assoc. for Computing Machinery, New York, NY, 303-310.

[12] Payne BA, Toga AW, "Distance field manipulation of surface models*", IEEE Comput Graph Appl* 121992, pp.65-71.

[13] C. W. Niblack, P.B. Gibbons, D. W. Capson, "Generating Skeletons and Centerlines from the Distance Transform," *CVGIP,* vol. 54 no. 5, sept. 1992, pp. 420-437.

[14] N. Gagvani, "Skeleton and Volume Thinning in Visualization" *MS Thesis,* Dept. of Electrical and Computer Engineering, Rutgers Univ., New Brunswick, N.J. June, 1997

[15] J.L. Helman, L. Hesselink, "Visualization of Vector Field Topology in Fluid Flows", *IEEE Computer Graphics and Application*, vol. 11, no.3, 1991, pp. 36-46.

[16] Y. Zhou, A.W. Toga, "Efficient Skeletonization of Volumetric Objects", *IEEE Trans on Visualization and Computer Graphics,* vol.5, no. 3, 1999, pp 196-209.

[17] F. Leymarie, M.D. Levine, "Simulating the Grass Fire Transform Using an Active Contour Model*", IEEE Trans on Pattern Analysis and Machine Intelligence,* vol.14, no. 1, Jan 1992, pp. 56-75.